Exam in Algorithms & Data Structures 3 (1DL481)

Prepared by Pierre Flener

Monday 12 March 2018 from 14:00 to 17:00, at Fyrislundsgatan 80, Sal 1

Materials: This is a ***closed***-book exam, drawing from the book *Introduction to Algorithms* by T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, published in 3rd edition by the MIT Press in 2009, and denoted by CLRS3 below. The usage of electronic devices is ***not*** allowed.

Instructions: Question 1 is ***mandatory***: you must earn ***at least half*** its points in order to pass the exam. Your answers must be written in English. Unreadable, unintelligible, and irrelevant answers will not be considered. Provide only the requested information and nothing else, but always show ***all*** the details of your reasoning, unless explicitly not requested, and make explicit ***all*** your assumptions. Answer each question only on the indicated pages. Do ***not*** write anything into the following table:

Question	Max Points	Your Mark
1	7	
2	3	
3	4	
4	6	
Total	20	

Help: Unfortunately, no teacher can attend this exam.

Grading: Your grade is as follows when your exam mark is e points, including ***at least*** 3.5 points on Question 1:

Grade	Condition
5	$18 \leq e \leq 20$
4	$14 \leq e \leq 17$
3	$10 \leq e \leq 13$
U	$00 \leq e \leq 09$

Identity: Your anonymous exam code:

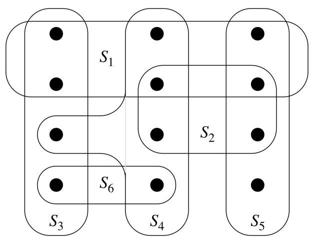
Answer to Question 1:

Question 1: NP-Completeness (*mandatory question*!) (7 points)

Do ***one*** of the following exercises from CLRS3 and earn ***at least half*** its points:

A. Exercise 34.5-6: A ***Hamiltonian path*** in a graph $G = (V, E)$ is a simple path that visits every vertex in V exactly once. The ***Hamiltonian-path problem*** asks whether a given graph contains a Hamiltonian path from a given vertex s to a given vertex t. Show that the Hamiltonian-path problem is NP-complete. (*AD3 teacher's hint*: Reduce from the Hamiltonian-*cycle* problem, which, given a graph $G' = (V', E')$, asks whether there is a simple cycle that contains each vertex in V' exactly once.)

B. Exercise 35.3-2: An instance $(X, \mathcal{F})$ of the ***set-covering problem*** consists of a finite set X and a family $\mathcal{F}$ of subsets of X, such that every element of X belongs to at least one subset in $\mathcal{F}$: $X = \bigcup_{S_i \in \mathcal{F}} S_i$. We say that a subset $S_i \in \mathcal{F}$ ***covers*** its elements. The problem is to find a minimum-size subset $\mathcal{C} \subseteq \mathcal{F}$ whose members cover all of X: $X = \bigcup_{S_i \in \mathcal{C}} S_i$.



To the left is a set-covering instance $(X, \mathcal{F})$, where X consists of the 12 black points and $\mathcal{F} = \{S_1, S_2, S_3, S_4, S_5, S_6\}$. A minimum-size set cover is $\mathcal{C} = \{S_3, S_4, S_5\}$, with size 3.

Show that the decision version of the set-covering problem is NP-complete by reducing it from [the decision version of] the ***vertex-cover problem***, which asks to find a minimum-size vertex cover in a given undirected graph $G = (V, E)$, that is a minimum-size subset $V' \subseteq V$ such that if $(u, v) \in E$, then either $u \in V'$ or $v \in V'$ (or both).

C. Problem 35-1a: ***Bin packing***: Suppose we are given a set of n objects, where the size s_i of the ith object satisfies $0 < s_i < 1$. We wish to pack all the objects into the minimum number of unit-size bins. Each bin can hold any subset of the objects whose total size does not exceed 1. Prove that the problem of determining the minimum number of bins required is NP-hard. (*Hint*: Reduce from the ***subset-sum problem***, which, given a finite set X of positive integers and an integer ***target*** $t > 0$, asks whether there is a subset $X' \subseteq X$ whose elements sum to t. For example, if $X = \{1, 2, 7, 32, 56, 35, 234, 12332\}$ and $t = 299$, then the subset $X' = \{2, 7, 56, 234\}$ is a solution, but there is no solution for $t = 11$.)

Chosen exercise: ....

Continued answer (you should start on the previous side):

Answer to Question 2:

Question 2: Probabilistic Analysis, Randomised Algorithms (3 points)

Do ***one*** of the following exercises from CLRS3:

A. Exercise 5.2-4: Use indicator random variables to solve the following problem, which is known as the ***hat-check problem***. Each of n customers gives a hat to a hat-check person at a restaurant. The hat-check person gives the hats back to the customers in a random order. What is the expected number of customers who get back their own hat? Are your indicator random variables independent?

B. Exercise 5.3-3: Many randomised algorithms randomise the input by permuting the given input array. Suppose that instead of swapping element $A[i]$ with a random element from the subarray $A[i \,.\,.\, n]$, we swapped it with a random element from anywhere in the array:

```
PERMUTE-WITH-ALL($A$)
1  $n = A.length$
2  for $i = 1$ to $n$            // RANDOM($\ell, u$) returns a random number from $\ell$ to $u$
3      swap $A[i]$ with $A[$RANDOM$(1, n)]$
```

Does this code produce a uniform random permutation? Why or why not?

Chosen exercise: ....

Continued answer (you should start on the previous side):

Answer to Question 3:

Question 3: Amortised Analysis (4 points)

Do the following exercise from CLRS3:

Exercise 17.2-1: Suppose we perform a sequence of stack operations (PUSH or POP) on a stack whose size never exceeds k. After every k operations, a COPY operation is invoked automatically to make a copy of the entire stack for backup purposes. Use an *accounting method of analysis* to show that the cost of n stack operations, including copying the stack, is $\mathcal{O}(n)$ by assigning suitable amortised costs to the various stack operations.

Continued answer (you should start on the previous side):

Answer to Question 4:

Question 4: Approximation Algorithms (6 points)

Do ***one*** of the following exercises from CLRS3:

A. Problem 35-1: ***Bin packing***: Suppose we are given a set of n objects, where the size s_i of the ith object satisfies $0 < s_i < 1$. We wish to pack all the objects into the minimum number of unit-size bins. Each bin can hold any subset of the objects whose total size does not exceed 1. The ***first-fit heuristic*** takes each object in turn and places it into the first bin that can accommodate it. Let $S = \sum_{i=1}^{n} s_i$.

   b. Argue that the optimal number of bins required, to be denoted by B^*, is at least $\lceil S \rceil$.

   c. Argue that the first-fit heuristic leaves at most one bin less than half full. Denote by B the number of bins used by the first-fit heuristic.

   e. Prove an approximation ratio of 2 for the first-fit heuristic.

B. Problem 35-3: ***Weighted set-covering problem***: Suppose that we generalise the set-covering problem (see Question 1B) so that each set S_i in the family $\mathcal{F}$ has an associated weight w_i and the ***weight*** of a cover $\mathcal{C}$ is $\sum_{S_i \in \mathcal{C}} w_i$. We wish to determine a minimum-weight cover. Section 35.3 handles the unweighted case in which $w_i = 1$ for all i, giving the following greedy set-covering heuristic:

```
GREEDY-SET-COVER($X, \mathcal{F}$)
1  $U = X$          // the set $U$ maintains the set of remaining uncovered elements
2  $\mathcal{C} = \emptyset$          // the set $\mathcal{C}$ maintains the cover being constructed
3  while $U \neq \emptyset$
4      select an $S_i \in \mathcal{F}$ that maximises $|S_i \cap U|$
5      $U = U - S_i$
6      $\mathcal{C} = \mathcal{C} \cup \{S_i\}$
7  return $\mathcal{C}$
```

   On the instance of Question 1B, this heuristic produces a sub-optimal cover of size 4 by selecting either the sets S_1, S_4, S_5, and S_3 or the sets S_1, S_4, S_5, and S_6, in order. Show how to generalise this heuristic in a natural manner to provide an approximate solution for any instance of the weighted set-covering problem. Does the CLRS3 analysis for the unweighted case still carry through, establishing that the generalised heuristic also is a polynomial-time $(\ln |X| + 1)$-approximation algorithm? [A yes/no answer with a high-level argument suffices for the last task.]

Chosen exercise: ....

Continued answer (you should start on the previous side):

Spare page for answers (or nice cartoons!)